In the Claims:

This listing of claims will replace all prior version and listings of claims in the application:

1. (Currently Amended) A computing machine, comprising:

first and second parallel buffers respectively associated with first and second data-processing units;

a processor coupled to the buffers and operable to[[,]]:

execute an application and first, second, third, and fourth data-transfer objects,

publish data under the control of the application,

load at least a portion of the published data into the first buffer under the control of the first data-transfer object,

load at least the same portion of the published data into the second buffer under the control of the second data-transfer object, and

retrieve at least the portion of the published data from the first and second buffers under the control of the third and fourth data-transfer objects, respectively.

2. (previously presented) The computing machine of claim 1 wherein:

the first and third data-transfer objects respectively comprise first and second instances of first object code; and

the second and fourth data-transfer objects respectively comprise first and second instances of second object code.

3. (previously presented) The computing machine of claim 1 wherein the processor comprises:

a processing unit operable to execute the application and to publish the data under the control of the application; and

a data-transfer handler operable to execute the first, second, third, and fourth data-transfer objects, to load the published data into the first and second buffers under

the control of the first and second data-transfer objects, respectively, and to retrieve the published data from the first and second buffers under the control of the third and fourth data-transfer objects, respectively.

4. (original) The computing machine of claim 1 wherein the processor is further operable to execute a thread of the application and to publish the data under the control of the thread.

5. (previously presented) The computing machine of claim 1 wherein the processor is further operable to:

execute a queue object and a reader object;

store a queue value under the control of the queue object, the queue value reflecting the loading of the published data into the first buffer;

read the queue value under the control of the reader object;

notify the third data-transfer object that the published data occupies the first buffer under the control of the reader object and in response to the queue value; and

retrieve the published data from the first buffer under the control of the third data-transfer object and in response to the notification.

6. (previously presented) The computing machine of claim 1, further comprising:

a bus; and

wherein the processor is operable to execute a communication object and to drive the data retrieved from one of the first and second buffers onto the bus under the control of the communication object.

7. (previously presented) The computing machine of claim 1, further comprising:

a third buffer; and

wherein the processor is operable to provide the data retrieved from one of the first and second buffers to the third buffer under the control of the respective one of the third and fourth data-transfer objects.

8. (previously presented) The computing machine of claim 1 wherein the processor is further operable to generate a message that includes a header and the data retrieved from one of the first and second buffers under the control of the respective one of the third and fourth data-transfer objects.

9. (previously presented) The computing machine of claim 1 wherein:

the first and third data-transfer objects respectively comprise first and second instances of first object code;

the second and fourth data-transfer objects respectively comprise first and second instances of second object code; and

the processor is operable to execute an object factory and to generate the first object code and the second object code under the control of the object factory.

10. Cancelled

11. (Currently Amended) A computing machine, comprising:

a first buffer;

a processor coupled to the buffer and operable to:

execute first and second data-transfer objects and an application,

generate data under the control of the application without generating an address of a destination of the data,

retrieve the generated data from the application and load the retrieved data into the buffer under the control of the first data-transfer object,

unload the data from the buffer under the control of the second data-transfer object, and

process the unloaded data under the control of the application without receiving the address of the destination of the data; and

~~The computing machine of claim 10~~ wherein the first and second data-transfer objects respectively comprise first and second instances of the same object code.

12. (Currently Amended) The computing machine of claim [[10]]11 wherein the processor comprises:

a processing unit operable to execute the application, generate the data, and process the unloaded data under the control of the application; and

a data-transfer handler operable to execute the first and second data-transfer objects, to retrieve the data from the application and load the data into the buffer under the control of the first data-transfer object, and to unload the data from the buffer under the control of the second data-transfer object.

13. (Currently amended) The computing machine of claim [[10]]11 wherein the processor is further operable to:

execute first and second threads of the application;

generate the data under the control of the first thread; and

process the unloaded data under the control of the second thread.

14. (Currently Amended) The computing machine of claim [[10]]11 wherein the processor is further operable to:

execute a queue object and a reader object;

store a queue value under the control of the queue object, the queue value reflecting the loading of the retrieved data into the first buffer;

read the queue value under the control of the reader object;

notify the second data-transfer object that the retrieved data occupies the buffer under the control of the reader object and in response to the queue value; and

unload the retrieved data from the buffer under the control of the second data-transfer object and in response to the notification.

15. (Currently amended) The computing machine of claim [[10]]11, further comprising:

a second buffer; and

wherein the processor is operable to execute a third data-transfer object, to unload the data from the first buffer into the second buffer under the control of the second data-transfer object, and to provide the data from the second buffer to the application under the control of the third data-transfer object.

16. Cancelled

17. (Currently Amended) The computing machine of claim [[10]]11 wherein:

the first and second data-transfer objects respectively comprise first and second instances of the same object code; and

the processor is operable to execute an object factory and to generate the object code under the control of the object factory.

18. (Currently Amended) The computing machine of claim [[10]]11 wherein the processor is further operable to package the generated data into a message that includes a header and the data under the control of the second data-transfer object.

19. (previously presented) A peer-vector machine, comprising:

a buffer;

a bus;

a processor coupled to the buffer and to the bus and operable to;

execute an application, first and second data-transfer objects, and a communication object,

publish data under the control of the application,

load the published data into the buffer under the control of the first data-transfer object,

retrieve the published data from the buffer under the control of the second data-transfer object,

construct a message under the control of the second data-transfer object, the message including the retrieved published data and information indicating a destination of the retrieved published data, and

drive the message onto the bus under the control of the communication object; and

a pipeline accelerator coupled to the bus, including the destination, and operable to receive the message from the bus, to recover the received published data from the message, to provide the recovered data to the destination, and to process the recovered data at the destination without executing a program instruction.

20. (previously presented) The peer-vector machine of claim 19 wherein the destination includes a field-programmable gate array that is operable to process the recovered data.

21. (previously presented) The peer-vector machine of claim 19, further comprising:

a registry coupled to the processor and operable to store object data; and

wherein the processor is operable to;

execute an object factory, and

generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory.

22. (Currently Amended) A peer-vector machine, comprising:

a buffer;

a bus;

a pipeline accelerator coupled to the bus and operable to generate data without executing a program instruction, to generate a header including information indicating a destination of the data, to package the data and header into a message, and to drive the message onto the bus; and

a processor coupled to the buffer and to the bus and operable to[[;]]:

execute an application, first and second data-transfer objects, and a communication object,

receive the message from the bus under the control of the communication object,

load into the buffer, under the control of the first data-transfer object, the received data without the header, the buffer corresponding to the destination of the data,

unload the data from the buffer under the control of the second data-transfer object, and

process the unloaded data under the control of the application.

23. (previously presented) The peer-vector machine of claim 22 wherein the processor is operable to:

receive the message from the bus under the control of the communication object; and

recover the data from the message under the control of the first data-transfer object.

24. (previously presented) The peer-vector machine of claim 22, further comprising:

a registry coupled to the processor and operable to store object data; and

wherein the processor is operable to,

execute an object factory, and

to generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory.

25. (withdrawn) A peer-vector machine, comprising:

a first buffer;

a bus;

a processor coupled to the buffer and to the bus and operable to,

execute a configuration manager, first and second data-transfer objects, and a communication object,

load configuration firmware into the buffer under the control of the configuration manager and the first data-transfer object,

retrieve the configuration firmware from the buffer under the control of the second data-transfer object, and

drive the configuration firmware onto the bus under the control of the communication object; and

a pipeline accelerator coupled to the bus and operable to receive the configuration firmware and to configure itself with the configuration firmware.

26. (withdrawn) The peer-vector machine of claim 25 wherein:

the processor is further operable to construct a message that includes the configuration firmware under the control of the second data-transfer object and to drive the message onto the bus under the control of the communication object; and

the pipeline accelerator is operable to receive the message from the bus and to recover the configuration firmware from the message.

27. (withdrawn) The peer-vector machine of claim 25, further comprising:

a registry coupled to the processor and operable to store configuration data; and

wherein the processor is operable to locate the configuration firmware from the configuration data under the control of the configuration manager.

28. (withdrawn) The peer-vector machine of claim 25, further comprising:

a second buffer; and

wherein the processor is operable to:

execute an application and third and fourth data-transfer objects,

generate a configuration instruction under the control of the configuration manager,

load the configuration instruction into the second buffer under the control of the third data-transfer object,

retrieve the configuration instruction from the second buffer under the control of the fourth data-transfer object, and

configure the application to perform an operation corresponding to the configuration instruction under the control of the application.

29. (withdrawn) The peer-vector machine of claim 25 wherein the processor is operable to:

generate a configuration instruction under the control of the configuration manager; and

configure the application to perform an operation corresponding to the configuration instruction under the control of the application.

30. (withdrawn) The peer-vector machine of claim 25 wherein the configuration manager is operable to confirm that the pipeline accelerator supports a configuration defined by the configuration data before loading the firmware.

31. (withdrawn) A peer-vector machine, comprising:

a first buffer;

a bus;

a pipeline accelerator coupled to the bus and operable to generate exception data and to drive the exception data onto the bus; and

a processor coupled to the buffer and to the bus and operable to,

execute an exception manager, first and second data-transfer objects, and an communication object,

receive the exception data from the bus under the control of the communication object,

load the received exception data into the buffer under the control of the first data-transfer object,

unload the exception data from the buffer under the control of the second data-transfer object, and

process the unloaded exception data under the control of the exception manager.

32. (withdrawn) The peer-vector machine of claim 31 wherein:

the pipeline is further operable to construct a message that includes the exception data and to drive the message onto the bus; and

the processor is operable to receive the message from the bus under the control of the communication object and to recover the exception data from the message under the control of the first data-transfer object.

33. (withdrawn) The peer-vector machine of claim 31, further comprising:

a second buffer;

wherein the processor is further operable to,

execute a configuration manager and third and fourth data-transfer objects,

generate configuration firmware under the control of the configuration manager in response to the exception data,

load the configuration firmware into the second buffer under the control of the third data-transfer object,

unload the configuration instruction from the second buffer under the control of the fourth data-transfer object, and

drive the configuration firmware onto the bus under the control of the communication object; and

wherein the pipeline accelerator is operable to receive the configuration firmware from the bus and reconfigure itself with the firmware.

34. (withdrawn) The peer-vector machine of claim 31 wherein the processor is further operable to:

execute an application and a configuration manager;

generate a configuration instruction under the control of the configuration manager in response to the exception data; and

reconfigure the application under the control of the application in response to the configuration instruction.

35. (withdrawn) A peer-vector machine, comprising:

a configuration registry operable to store configuration data;

a processor coupled to the configuration registry and operable to locate configuration firmware from the configuration data; and

a pipeline accelerator coupled to the processor and operable to configure itself with the configuration firmware.

36. (withdrawn) A peer-vector machine, comprising:

a configuration registry operable to store configuration data;

a pipeline accelerator; and

a processor coupled to the configuration registry and to the pipeline accelerator and operable to retrieve configuration firmware in response to the configuration data and to configure the pipeline accelerator with the configuration firmware.

37. (previously presented) A method, comprising:

publishing data with an application that does not generate an address of a destination of the data;

loading the published data into a first buffer with a first data-transfer object, the loaded data absent the address of the destination of the data, each location within the buffer corresponding to the address;

retrieving the published data from the buffer with a second data-transfer object, the retrieved data absent the address;

generating a message header that includes the address; and

generating a message that includes the retrieved data and the message header.

38. (original) The method of claim 37 wherein publishing the data comprises publishing the data with a thread of the application.

39. (previously presented) The method of claim 37, further comprising:

generating a queue value that corresponds to the presence of the published data in the buffer;

notifying the second data-transfer object that the published data occupies the buffer in response to the queue value; and

wherein retrieving the published data comprises retrieving the published data from the buffer with the second data-transfer object in response to the notification.

40. (previously presented) The method of claim 37, further comprising driving the message onto a bus with a communication object.

41. (original) The method of claim 37, further comprising loading the retrieved data into a second buffer with the second data-transfer object.

42. (currently amended) The method of claim 37 wherein ~~generating~~ the message header and the message comprises generating the message header and the message with the second data-transfer object.

43. (original) The method of claim 37, further comprising:

generating data-transfer object code with an object factory;

generating the first data-transfer object as a first instance of the object code; and

generating the second data-transfer object as a second instance of the object code.

44. (previously presented) The method of claim 37, further comprising receiving the message and processing the data in the message with a hardwired pipeline accelerator.

45. Cancelled.

46. (Currently Amended) The method of claim [[45]]47 wherein processing the unloaded data comprises processing the unloaded data with a thread of the software application.

47. (Currently Amended) A method, comprising:

receiving a message that includes data and that includes a message header that indicates a destination address of the data, the destination address corresponding to a software application;

loading into a first buffer, with a first data-transfer object, the received data without the message header, the first buffer corresponding to the destination;

unloading the data from the buffer with a second data-transfer object;

processing the unloaded data with the software application without the software application receiving the destination address of the data;~~The method of claim 45, further comprising:~~

generating a queue value that corresponds to the presence of the data in the buffer;

notifying the second data-transfer object that the data occupies the buffer in response to the queue value; and

wherein unloading the data comprises unloading the data from the buffer with the first data-transfer object in response to the notification.

48. (Currently Amended) The method of claim [[45]]47 wherein receiving the message comprises receiving the message with the first data-transfer object.

49. (Currently Amended) The method of claim [[45]]47, further comprising:

wherein receiving the message comprises retrieving the message from a bus with a communication object; and

transferring the data from the communication object to the first data-transfer object.

50. (Currently Amended) The method of claim [[45]]47, further comprising generating the message header and the message with a hardwired pipeline accelerator.

51. (previously presented) A method, comprising:

publishing data with an application running on a processor;

loading the published data into a buffer with a first data-transfer object running on the processor;

retrieving the published data from the buffer with a second data-transfer object running on the processor;

generating information that indicates a hardwired pipeline for processing the retrieved data;

packaging the retrieved data and the information into a message;

driving the message onto a bus with a communication object running on the processor;

receiving the message from the bus; and

processing the published data with the indicated hardwired pipeline without executing a program instruction, the indicated hardwired pipeline being part of a pipeline accelerator that includes a field-programmable gate array.

52. (previously presented) The method of claim 51 wherein:

packaging the retrieved data and the information into a message comprises generating the message including a header, and the published data, with the second data-transfer object;

driving the data onto the bus comprises driving the message onto the bus with the communication object; and

receiving the published data comprises receiving the message and recovering the published data from the message with the pipeline accelerator.

53. (previously presented) A method, comprising:

generating, with a pipeline accelerator and without executing a program instruction, a message header that includes a destination of data, the destination identifying a software application for processing the data;

generating, with the pipeline accelerator and without executing a program instruction, a message that includes the header and the data;

driving the message onto a bus with the pipeline accelerator;

receiving the message from the bus with a communication object running on a processor;

loading into a buffer, with a first data-transfer object running on the processor, the received data absent the header, the buffer being identified by the destination;

unloading the data from the buffer with a second data-transfer object running on the processor; and

processing the unloaded data with the software application running on the processor.

54. (previously presented) The method of claim 53, further comprising recovering the data from the message with the first data-transfer object.

55. (withdrawn) A method, comprising:

retrieving configuration firmware with a configuration manager;

loading the configuration firmware into a first buffer with a first communication object;

retrieving the configuration firmware from the buffer with a second communication object;

driving the configuration firmware onto a bus with an communication object;

receiving the configuration firmware with a pipeline accelerator; and

configuring the pipeline accelerator with the configuration firmware.

56. (withdrawn) The method of claim 55, further comprising:

generating a configuration instruction with the configuration manager; and

configuring the application to perform an operation corresponding to the configuration instruction.

57. (withdrawn) The method of claim 55, further comprising:

generating a configuration instruction with the configuration manager;

loading the configuration instruction into a second buffer with a third communication object;

retrieving the configuration instruction from the second buffer with a fourth communication object; and

configuring the application to perform an operation corresponding to the configuration instruction.

58. (withdrawn) A method, comprising:

generating exception data and driving the exception data onto a bus with a pipeline accelerator;

receiving the exception data from the bus with a communication object;

loading the received exception data into a buffer with a first data-transfer object;

unloading the exception data from the buffer with a second data-transfer object; and

processing the unloaded exception data under with an exception manager.

59. (withdrawn) The method of claim 58, further comprising:

retrieving configuration firmware with a configuration manager in response to the exception data,

loading the configuration firmware into a second buffer with a third transfer object;

unloading the configuration instruction from the second buffer with a fourth data-transfer object;

driving the configuration firmware onto the bus with the communication object; and

reconfiguring the pipeline accelerator with the configuration firmware.

60. (withdrawn) The method of claim 58, further comprising:

generating a configuration instruction with a configuration manager in response to the error data; and

reconfiguring the application in response to the configuration instruction.

61. (withdrawn) A method, comprising:

retrieving configuration firmware pointed to by configuration data stored in a configuration registry during an initialization of a computing machine; and

configuring a pipeline accelerator of the computing machine with the configuration firmware.

62. (previously presented) A peer-vector machine, comprising:

a buffer;

a single bus;

a processor coupled to the buffer and to the bus and operable to[[;]]:

execute an application, first and second data-transfer objects, and a communication object,

publish data under the control of the application,

load the published data into the buffer under the control of the first data-transfer object,

retrieve the published data from the buffer under the control of the second data-transfer object,

construct a message under the control of the second data-transfer object, the message including the retrieved published data and information indicating a destination of the retrieved published data, and

drive the message onto the bus under the control of the communication object; and

a pipeline accelerator coupled to the bus, including the destination, and operable to receive the message from the bus, to recover the received published data from the message, to provide the recovered data to the destination, and to process the recovered data at the destination without executing a program instruction.